Witam, witam.
Kłaniam się. Wydawać by się mogło, że po vertex shaderach nie będziemy się musieli już niczego uczyć. Wiemy takie rzeczy, że możemy robić co tylko chcemy. I po części jest to prawda. Vertex shader do spółki z pixel shader'em umożliwia tworzenie z naszej wyobraźni najwymyślniejszych cudów. Ale mają one oczywiście także wady. Po pierwsze - ilość instrukcji, która skutecznie ograniczy nam w pewnym momencie nasze możliwości. Po drugie robienie wszystkiego na bardzo niskim poziomie - praktycznie rejestrów i asemblera. Dlatego też naszej nauki nie możemy zakończyć w tym miejscu. Jak przekonamy się dzisiaj to dopiero początek tego wielkiego świata, jakim jest grafika komputerowa i aby w nim się nie zgubić trzeba się jeszcze dużo dowiedzieć. Zatem dzisiaj wyczekiwana zapewne z dużą niecierpliwością pierwsza część opowieści o mapowaniu środowiskowym czyli rzecz o współrzędnych sferycznych - łykniemy trochę wiadomości o tworzeniu map świata oraz pobawimy się znowu matematyką.

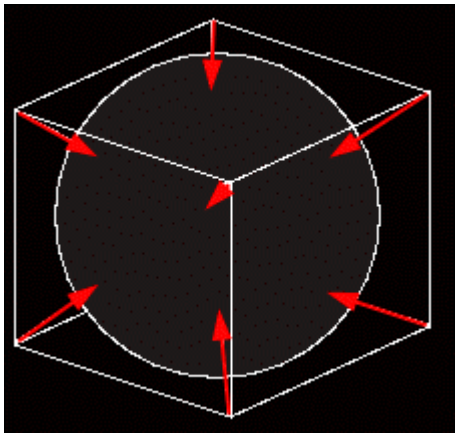
Pewnie wielu z was ma już dosyć (pomimo tego, że i tak robią wrażenie) naszych brył oświetlonych i poteksturowanych. Są takie jakie są ale ciągle mamy wrażenie, że czegoś im jeszcze brakuje do pełni szczęścia. Kłuje w oczy coś, co nie pozwala nam tak do końca uwierzyć. Są one jakieś takie zbyt idealne, zbyt kolorowe. Cóż więc zrobić aby to zmienić? Nic prostszego - dodać trochę więcej realizmu. Ktoś powie "Phi! to wiemy, ale jak to zrobić?". Pomyślmy przez chwilę i, co chyba nawet ważniejsze, poobserwujmy to, co pragniemy odwzorować na ekranie. Wprawdzie grafika komputerowa służy raczej do tworzenia światów wirtualnych, no ale nic nie jest lepszym wzorcem dla świata wymyślonego niż ten prawdziwy. A to, że będzie on bardziej podobny do tego zwykłego, szarego może tylko polepszyć nasze wrażenia. I cóż takiego widzimy kiedy obserwujemy naszą rzeczywistość? Każdy zapewne w tym momencie widzi co innego wokół siebie i co innego pragnie odwzorować. Ale wszystko co chcemy zrobić odbywa się w jakimś określonym miejscu. Miejsce to ma określony wygląd, światło i przedmioty się tam znajdujące. Wszystko to razem tworzy scenę, którą my pragniemy odwzorować na naszym monitorze. Jeśli zamodelujemy wszystkie nasze przedmioty, przeniesiemy je do naszego świata wirtualnego, nałożymy tekstury i oświetlimy wszystko będzie w jaki najlepszym porządku, ale braknie nam czegoś bliżej nieokreślonego. Wszystko będzie takie suche, sztywne, niemal plastykowe. Trzeba więc wnieść na scenę trochę życia! A co wnosi na scenę życie? Światło! Ale nie będzie to światło w sensie, który my już znamy. Na naszej scenie mamy światełka, oświetlamy pięknie wszystkie nasze obiekty. Mówię tutaj o świetle, które będzie sobie wędrowało po scenie, odbijało się od różnych obiektów i dopiero na samym końcu trafi do naszego oka. To, że się odbija po drodze od wszystkiego, co tylko napotka spowoduje, że zostanie ono w różnym stopniu pochłonięte przez przedmioty, które znajdzie na swojej drodze. W efekcie trafi do naszego oka coś zupełnie nieokreślonego, czego się nie da przewidzieć prostą symulacją i obliczeniami kolorów, jakie czyniliśmy do tej pory. Oczywiście niemożliwe jest obliczenie rzeczywistych odbić i pochłonięć przez dzisiejsze komputery - nie wiem nawet czy w ogóle będzie to kiedykolwiek możliwe. Dlatego, jak w całej grafice komputerowej, tak i w tym przypadku posłużymy się pewnym przybliżeniem. Zauważmy za to, że każde kolejne przybliżenie coraz bardziej urealnia nasz świat i będzie on coraz bardziej przypominał ten prawdziwy, jaki znamy z dnia powszedniego. Wiecie zapewne o czym mowa, więc, żeby nie przeciągać pomówmy o tym czymś.

Mapowanie środowiskowe - to inaczej technologia, która będzie nam pozwalała na symulowanie odbić światła od powierzchni. Dlaczego nazywa się mapowaniem środowiskowym? W zasadzie odpowiedziałem wyżej na to pytanie, no ale wyjaśnijmy już sobie to tak do końca. Wiemy, że do naszego oka trafia tylko to, co zostanie odbite (nie pochłonięte) przez materiał, z jakiego zbudowany jest obiekt. Jeśli promień, który przyleci skądś zacznie się odbijać od różnych przedmiotów na naszej scenie to straci na pewno cześć koloru. Do naszego oka trafi taki, że będzie zawierał tylko to, co nie zostało mu zabrane po drodze. To co zostanie, utworzy nam obraz, który my możemy oglądać. Ale w wielu przypadkach nie będzie to obraz jednego obiektu! Przecież promień odbijał się od wielu po kolei, na każdym tracąc część informacji. W wyniku tego do naszego oka trafia obraz, który doskonale znamy z rzeczywistego świata. Na obiektach, które oglądamy na scenie pojawiają się odbicia innych obiektów, otaczających je. Dla danego, obserwowanego szczególnie przez nas obiektu wszystko inne jest środowiskiem, w którym on sam przebywa - stąd też zapewne nazwa - "mapowanie środowiskowe". Oczywiście pojawianie się odbić będzie ściśle związane z rodzajem powierzchni, jaką posiada dany obiekt. Jeśli obserwować będziemy na przykład dywan lub ścianę to na pewno na nich takich odbić nie zobaczymy - obiekty te są w naszym rozumieniu "matowe". W rozumieniu światła mają one zbyt nieregularną powierzchnię aby móc odbić światło w taki sposób, żebyśmy mogli zobaczyć cokolwiek odbijającego się od takiego rodzaju powierzchni. Ale wystarczy spojrzeć na kineskop naszego monitora, jakąkolwiek szklankę stojącą na stole czy blat polerowanego stołu. Choć niewyraźnie to jednak widać, że coś tam się odbija. Dzisiejsza lekcja nie będzie służyła nauce robienia luster - one odbijają wszystko bardzo dokładnie, natomiast jak widać na naszych przykładach, na podstawie tego co się odbija od naszych obiektów trudno jest wnioskować o kształcie otaczającego świata. I też nie do końca o to chodzi. My będziemy tylko chcieli zasugerować widzowi, który oglądał będzie nasze sceny, że pewne obiekty są zbudowane z materiałów, które potrafią odbijać mnóstwo padającego na nie światła - takich jak metal, szkło, powierzchnie polerowane czy woda. Jak napisałem wcześniej policzenie wszystkich odbić poszczególnych promieni padających na scenę jest dzisiaj zupełnie niemożliwe w rozsądnym czasie, dlatego my posłużymy się kolejną sztuczką stosowaną w "szybkiej" grafice 3D.
Na czym będzie polegało mapowanie środowiskowe? Otóż, jak wiele innych rzeczy jest to banalnie proste ;). Zamiast liczyć wszystkie promienie aby pomalować odpowiednio nasz obiekt, my zrobimy tak: Określimy sobie co będzie widać z punktu położenia naszego obiektu (tak, jak my byśmy byli na jego miejscu), potem zrobimy sobie z tego widoku teksturę, trochę nią pomieszamy i na końcu nałożymy na nasz obiekt w pewien specyficzny sposób. Metoda taka ma oczywiście pewne zalety i pewne wady. Zaletą niewątpliwie jest jej duża szybkość. Jak wiemy nakładanie wielu tekstur na obiekty nasz sprzęt może wykonać w bardzo fajny sposób (patrz lekcje o multiteksturingu). Wystarczy, że zmieszamy kolor naszego obiektu, teksturę, która jest na niego nałożona i teksturę reprezentującą świat otaczający nasz obiekt. Mapa środowiskowa w większości przypadków powinna całkowicie wystarczyć. Jeśli przyjrzeć się odbijającym otoczenie przedmiotom w rzeczywistym

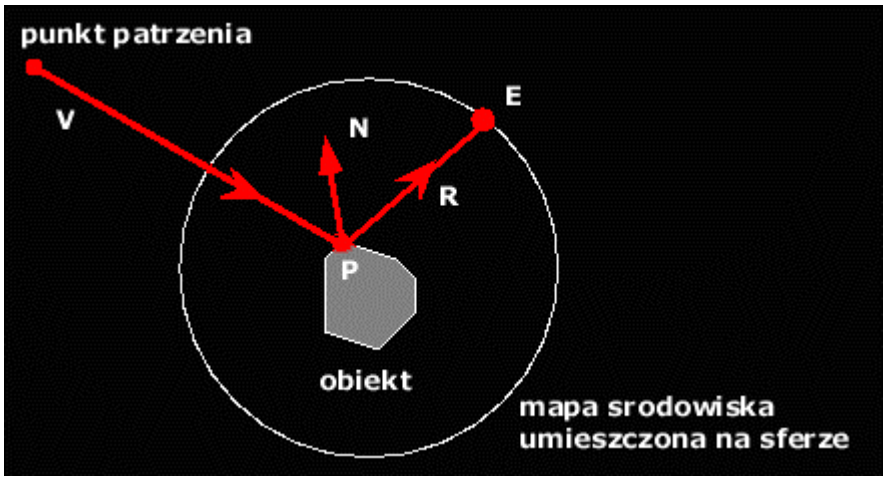
świecie, to większość z nich nie jest zupełnie płaska. Powoduje to, że nasza mapa odzwierciedlająca środowisko nie musi być bardzo dokładna; wystarczy, że będzie mniej więcej prezentować to, co powinno być widoczne. Oczywiście im obiekt większy, bardziej płaski czy bardziej odbijający światło tym nasza tekstura musi być dokładniejsza. Niestety sposób ten nie jest pozbawiony wad. Podstawowym problemem będzie oczywiście ruch. No bo co się stanie jeśli nagle przedmioty otaczające nasz obiekt (ten, w którym się wszystko odbija) zaczną się poruszać? Oczywiście naszą mapę możemy wyrzucić do śmieci, bo nie będzie już odzwierciedlać tego, co otacza nasz obiekt. Ale nie możemy wpadać w panikę. Nie jest tak źle jak to wygląda na pierwszy rzut oka. W zależności od tego na czym nam najbardziej zależy będziemy mogli radzić sobie w różnorakie sposoby. Istnieje wiele sposobów "robienia" mapowania środowiskowego, jednak my poznamy dwa - jedną metodę prostszą, drugą trochę bardziej zakręconą. Na pierwszy ogień pójdzie więc metoda prostsza i aktualnie szerzej znana, choć powoli wypierana przez tę drugą. Mapowanie środowiskowe podzielimy sobie ze względu na sposób, w jaki będziemy tworzyć nasze mapy środowiskowe i jaki będziemy obliczać współrzędne dla mapy na obiekcie. Powiemy sobie więc o mapowaniu **sferycznym** i **sześciennym**. Jak zapewne słusznie zauważyliście tylko te dwie metody opisane są w dokumentacji do SDK i na tym opisie oraz na naszym przykładzie oprę dzisiejsze rozważania. Dzisiaj mowa będzie o **mapowaniu sferycznym**.
Czym ono się charakteryzuje? Otóż tekstura, czy też mapa (będziemy w tym przypadku używać tych określeń zamiennie) będzie reprezentowała rzeczywistość otaczającą nasz obiekt w pewien specyficzny sposób. Stojąc w miejscu obiektu możemy rozglądać się dookoła. Wprawdzie widzimy wszystko jako 3D i trudno jest nam sobie wyobrazić to sobie jako płaski obraz to jednak możemy zrobić zdjęcia. Aby sfotografować całą otaczającą nas rzeczywistość będziemy musieli obrócić się we wszystkie możliwe strony. Jeśli teraz zbudujemy sobie dla przykładu kulę i jej wnętrze wyłożymy naszymi odbitkami (oczywiście każdą we właściwym miejscu) a następnie wejdziemy do niej, to będziemy mogli odnieść wrażenie, że mamy jakąś tam namiastkę naszej rzeczywistości. Tak mniej więcej działa sferyczne mapowanie środowiska. Na obiekt nałożymy teksturę w taki sposób, że tekstura zostanie niejako "zrzutowana" z kuli otaczającej nasz obiekt na niego samego. Oczywiście żadnej kuli w programie tworzyć nie będziemy, natomiast stworzymy specjalnego rodzaju teksturę, która po odpowiednim dobraniu współrzędnych mapowania pozwoli się "nałożyć" na obiekt w sposób podobny do opisanego powyżej. W dzisiejszej lekcji poruszymy sobie dwie istotne i zupełnie niezbędne do dalszego działania sprawy. Pierwszą będzie sposób tworzenia map środowiska dla naszych potrzeb - dowiemy się jakich metod możemy użyć do stworzenia potrzebnej tekstury i jak się w ogóle do tego zabrać od bardziej praktycznej strony. Drugą sprawą będzie sposób obliczenia współrzędnych mapowania dla mapy świata na obiekcie podczas jego renderingu na ekranie. Zaczniemy może od ogólnego omówienia algorytmu mapowania i generowania współrzędnych.

Przy renderingu każdego wierzchołka naszej bryły będziemy musieli dla niego wyznaczyć takie współrzędne mapowania, których ustawienie na mapie środowiska spowoduje odpowiednie "odbicie" właściwego fragmentu świata zawartego w mapie. Dodatkowo trzeba to będzie robić za każdym przebiegiem pętli renderującej, ponieważ nasz obiekt będzie się mógł na przykład poruszać co automatycznie pociągnie za sobą zmianę tych współrzędnych. Aby policzyć współrzędne dla mapy świata potrzebne będą nam dwie rzeczy - współrzędne wierzchołków w układzie współrzędnych świata oraz ich normalne, również przedstawione w ten sam sposób. Naszym głównym celem stanie się wyznaczenie wektora refleksu, który następnie posłuży do wyznaczenia współrzędnych mapowania na mapie środowiska. Najpierw należy więc zadbać o to, aby otrzymać współrzędne wierzchołków. Współrzędne wierzchołków będą odpowiadać różnicy pomiędzy wektorem wyznaczającym położenie kamery a współrzędną wierzchołka. Mając taki wektor, znormalizujemy go , żeby wygodniej było prowadzić obliczenia. Następne co zrobimy to wyliczenie na podstawie tej współrzędnej wierzchołka i jego normalnej wektora refleksu, który nie będzie niczym innym jak sumą odwróconego wektora wskazującego wierzchołek we współrzędnych kamery i wektora normalnego. posługując się wektorem refleksu będziemy mogli pobrać sobie współrzędne na naszej mapie zawierającej środowisko.

Najpierw wyobraźmy sobie sytuację jak na rysunku - potrzebujemy sferę, na którą nałożony zostanie obraz całego otaczającego ją świata - oczywiście musi być ona umieszczona w punkcie, który odpowiada dokładnie położeniu naszego obiektu w późniejszym czasie na scenie. Będąc dokładnie w tym punkcie my nie będziemy mieć w ręce aparatu i poradzić będziemy musieli sobie w inny sposób. Tym sposobem będzie wyrenderowanie sześciu płaszczyzn otaczających naszą hipotetyczną kulę od góry, dołu, północy, południa wschodu i zachodu. Oczywiście położenie tych płaszczyzn nie musi się zgadzać z kierunkami geograficznymi, ale muszą być do siebie wzajemnie prostopadłe aby stworzyć sześcian. Obrazy umieszczone na poszczególnych ścianach muszą też w miarę dobrze przylegać do siebie aby na późniejszej mapie nie było większych zakłóceń. Każdy obiekt będzie wyrenderowany na omawianych płaszczyznach oczywiście poza tym, na który będziemy potem taką mapę nakładać.
Następnie obiekt, który będzie odzwierciedlał nasze środowisko zostanie umieszczony w punkcie, w którym uprzednio znajdowała się kula zawierająca odbicie środowiska a kamera znajdzie się w zakładanym punkcie obserwacji. Dla każdego wierzchołka naszego obiektu wektor łączący położenie kamery i jego samego zostanie odbity względem wektora normalnego w naszym wierzchołku, dając kierunek mapowania z przestrzeni 3D na mapę 2D. Całą sytuację przedstawiają poniższe rysunki:



Czerwone strzałki oznaczają kierunek rzutowania sześciu płaszczyzn zawierających obraz środowiska na kulę. Po takim zmapowaniu ostatnim krokiem będzie spłaszczenie obrazu kuli do mapy 2D.
Podczas renderowania obiektu z nałożoną mapą tekstury oczywistym faktem jest to, że te potrzebne do nałożenia mapy na obiekt muszą być wyliczane dla każdego wierzchołka. Współrzędne **u** i **v** są obliczane takim samym sposobem jak podczas obliczania i generowania mapy środowiska - poprzez mapowanie współrzędnych 3D do przestrzeni dwuwymiarowej. Współrzędne każdego wierzchołka mapowanego modelu i kierunki normalnych w tych wierzchołkach są używane do obliczenia wektorów refleksów. Wektor widoku jest przeważnie przedstawiany jako odległość od kamery (która znajduje się i tak w centrum układu we współrzędnych kamery) do poszczególnych wierzchołków. Przy takich założeniach oznaczmy sobie:



V - wektor widoku,
N - normalna w określonym wierzchołku w układzie współrzędnych kamery,
E - punkt na sferze z mapą środowiska, w który powinien trafić wektor refleksu,
wektor refleksu **R** przedstawia się wzorem:

$$d = V \cdot N$$

$$R = V - 2 \cdot d \cdot N$$

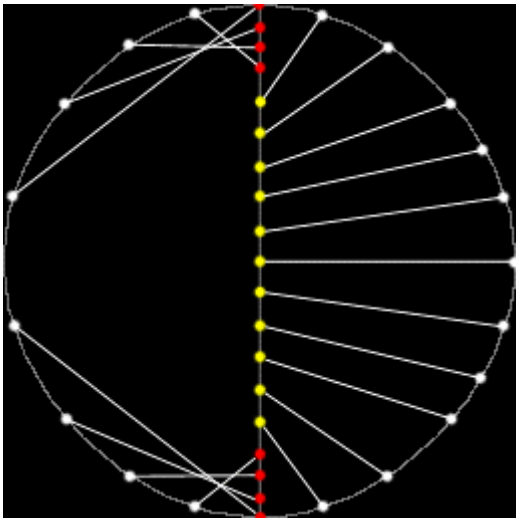
Wzór jest bardzo prosty do zapamiętania i nie sądzę, żeby były z nim jakieś problemy. Zresztą jak sobie rozrysujecie te wektory i pokombinujecie trochę to sami dojdziecie pewnie co z czego się wzięło, no a dla tych, którym się nie chce to jest gotowiec. Mając wektor refleksu możemy przystąpić do wyznaczenia, który piksel będzie szukanym na mapie świetlnej. Posłużą nam do tego trzy wzory:

$$m = 2 \cdot \sqrt{(R_x)^2 + (R_y)^2 + (R_z + 1)^2}$$

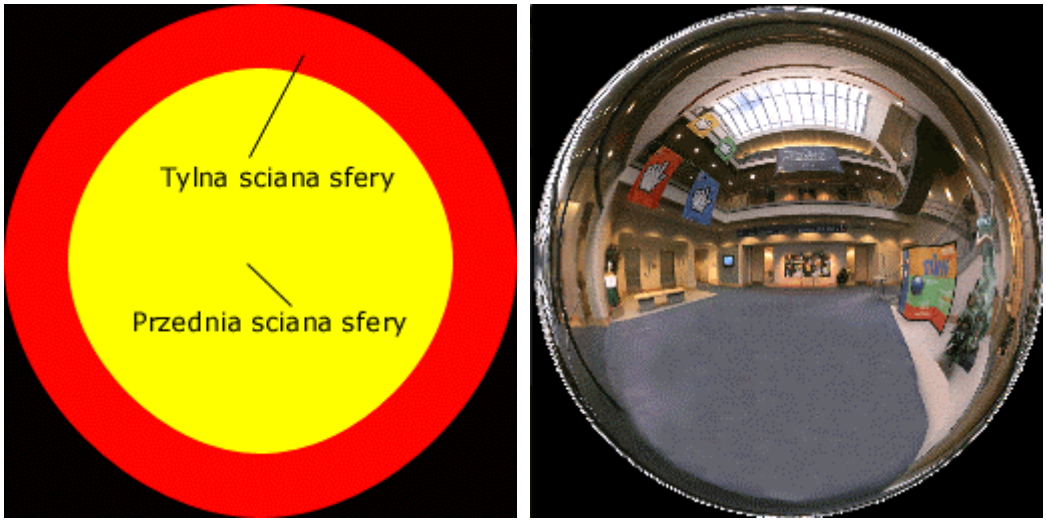
$$u = \frac{R_x}{m} + \frac{1}{2}$$

$$v = \frac{R_y}{m} + \frac{1}{2}$$

Można by długo się głowić skąd to się wzięło i na co, ale jak już mówiłem sposobów wyznaczania współrzędnych mapy środowiska jest wiele i ten stanowi tylko jeden z nich. Wzory te są ściśle związane ze sposobem, w jaki wygenerujemy mapę środowiska. Popatrzmy na poniższy rysunek:



Powyżej przedstawione przez nas wzory oznaczają, że cała powierzchnia sfery otaczającej nasz obiekt, na którym chcemy odwzorować środowisko będzie odwzorowana na naszej mapie, którą będzie de facto koło (na kwadratowej teksturze). Powyższy rysunek pokazuje jakie fragmenty naszej hipotetycznej sfery znajdą się w jakiej części mapy środowiska. W tym przykładzie oko znajduje się po prawej stronie. Białe punkty na powierzchni kuli są zrzucane do miejsc na mapie, którą symbolizują na powyższym rysunku punkty żółte i czerwone. Jak widać, wszystko co jest po prawej stronie kuli zostanie zrzucone niejako do wewnętrznej części mapy (w kolorze żółtym). Część widoczna z tyłu zostanie także uwzględniona, żeby było bardziej realistycznie - będzie ona stanowiło otoczkę wnętrza (kolor czerwony). Następny rysunek powinien wam już całkiem wyjaśnić sytuację (dla porównania umieściłem oryginalną mapę środowiska, aby uzmysłowić wam, o co chodzi):



Jeśli przyjrzeć się teraz jakiejś rzeczywistej, wygenerowanej przez kogoś mapie to da się dosyć prosto wywnioskować skąd i co się wzięło. Mam nadzieję, że to choć trochę pomoże wam zrozumieć ideę.

Samo mapowanie sferyczne może oczywiście być użyte do osiągnięcia innych efektów niż tylko odbicia lustrzane. Każdy efekt, który będzie się opierał na liczeniu różnych wektorów związanych z powierzchnią będzie można w jakiś tam sposób zasymulować właśnie za pomocą mapowania sferycznego - dla przykładu model cieniowania Phonga albo efekty refrakcji (załamywania się światła w różnych ośrodkach) czy określania koloru w jakiś niebanalny sposób. Jeśli użyjemy mapowania sferycznego do uzyskania jakichś interesujących efektów to oczywiście musimy pamiętać, że współrzędne tekstur dla naszych wymysłów będziemy musieli policzyć już jednak sami.

Wracając do samych map środowiska powiedzmy może sobie w skrócie jak można sobie je wygenerować do naszych

potrzeb. Sposobów tak naprawdę jest bardzo wiele, powiedzmy sobie może o kilku. Zacząć można od sposobów, że tak powiem "sprzętowych". Pierwszym, najzupełniej banalnym jest zrobienie zdjęcia chromowej kuli, która odzwierciedla wszystko dookoła. Zadanie zupełnie proste nawet dla laika, ma jednak bardzo poważną wadę - w kuli tej będzie się odbijać sam aparat, którym zrobimy zdjęcie, więc nie będzie ono nigdy idealnie odzwierciedlać naszego środowiska. W wielu przypadkach więc rozwiązanie to będzie nie wystarczające.
Innym sposobem jest użycie specjalnego typu obiektywu (tzw. "rybie oko"). Aparat z takim obiektywem potrafi zrobić zdjęcie przypominające nieco wyglądem teksturę potrzebną do mapowania środowiskowego, nie każdy jednak posiada takie cudo w swoich zbiorach. Ale pomimo tych wszystkich przeciwności mam dla was bardzo dobrą wiadomość - teksturę dla mapowanie sferycznego można również wygenerować programowo. Większość sposobów programowych, jeśli nie wszystkie opierają się pośrednio lub bezpośrednio na metodzie śledzenia promieni. Promienie refleksu (R) odbijają się od naszego obiektu i przynoszą nam informację ze środowiska. My jednak dla ułatwienia nieco sobie sprawy spójrzmy na to w drugą stronę - wyślijmy promienie od naszego oka do środowiska (poprzez obiekt), żeby określić to, co odbija się na naszej kuli i potem zostanie przekazane na naszą teksturę. Czyli promień biegnie od oka, odbija się od obiektu i leci gdzieś w przestrzeń, trafiając w końcu na coś, co może nas zainteresować z punktu widzenia obecności tego na mapie środowiska.
Nie najprostszym, a zarazem najbardziej obliczeniowo-żernym sposobem generowania mapy jest bezpośrednie wypuszczenie promieni w środowisko i sprawdzenie w co trafią. Prosta implementacja takiego algorytmu może zostać przedstawiona na kawałku pseudo kodu poniżej. Oczywiście procedura ta nie jest w żaden sposób zoptymalizowana, więc macie ewentualne pole do popisu.

```
void gen_sphere_map( int width, int height, float pos[3], float (*tex)[3] )
{
  float ray[3], color[3], p[3], s, t;
  int i, j;

  for (j = 0; j < height; j++)
  {
    t = 2.0 * ((float)j / (float)(height-1) - .5);
    for (i = 0; i < width; i++)
    {
      s = 2.0 * ((float)i / (float)(width - 1) - .5);
      if (s*s + t*t > 1.0)
        continue;

      /* compute the point on the sphere (aka the normal) */
      p[0] = s;
      p[1] = t;
      p[2] = sqrt(1.0 - s*s - t*t);

      /* compute reflected ray */
      ray[0] = p[0] * p[2] * 2;
      ray[2] = p[1] * p[2] * 2;
      ray[3] = p[2] * p[2] * 2 - 1;
      fire_ray(pos, ray, tex[j*width + i]);
    }
  }
}
```

Najbardziej interesująca część działań jakie przeprowadzamy za pomocą takiego działania jest zawarta w funkcji niewidocznej tutaj, czyli **fire_ray**. Funkcja ta oblicza przecięcia promieni biegnących od środka naszej kuli poprzez całe środowisko (coś w stylu algorymu kolizji jeśli miałbym porównywać). Jeśli promień w coś trafi wtedy pobierany jest kolor tego czegoś i jest on umieszczany w trzecim parametrze tej funkcji, który de facto będzie pikselem położonym w odpowiednim punkcie naszej mapy reprezentującej środowisko. Oczywiście sposób, który właśnie przedstawiłem nie będzie pozbawiony wad. Takie tworzenie mapy środowiska otaczającego nasz świat będzie powodować powstawanie różnych zakłóceń na teksturze i w konsekwencji niezbyt piękne rezultaty na ekranie. W rzeczywistości, aby osiągnąć zadowalające rezultaty należałoby poprowadzić kilka promieni w środowisko dla jednego teksela tekstury i potem pokombinować z rezultatami - na przykład przefiltrować wyniki dla jednego teksela wzajemnie ze sobą. Jest to jednak metoda zbyt kosztowna, nawet jak na możliwości współczesnych akceleratorów. Przecięcia naszych promieni ze środowiskiem i otrzymanie kolorów, które powstaną w odpowiednich miejscach mapy może być osiągnięte też trochę mniejszym nakładem pracy poprzez użycie jakiegoś pakietu z zaimplementowaną technologią śledzenia promieni i naszej sceny. Wstawiamy kulę na scenę i renderujemy na jej powierzchnię, wszystko co widać z jej wnętrza.
Innym sposobem, prostszym nieco od metody śledzenia promieni jest przedstawienie naszej sceny jako sześciu obrazów, formujących sześcian wokół punktu, dla którego aktualnie generujemy naszą mapę środowiska. Obrazki te będą reprezentować to co widzi aparat fotograficzny o kącie widzenia 90 stopni i punkcie centralnym widzenia aparatu umieszczonym w centrum takiego obrazka. Takie sześć obrazków może zostać bez problemu wygenerowanych za pomocą naszego pakietu graficznego, za pomocą jakiegoś programu do tworzenia grafiki trójwymiarowej a może po prostu powstać poprzez zrobienie takich zdjęć (jeśli ma to być coś realnego). Kiedy już będziemy mieli takie obrazki wtedy promienie pobiegną sobie przez kulę i sześcian ją otaczający, aby określić które teksele znajdą się w teksturze odwzorowującej nasze

środowisko. Promień biegnący od środka kuli, poprzez jej powierzchnię i przecinając otaczające ją ściany sześcianu określi nam dokładnie co powinno się znaleźć na powierzchni kuli. Omówmy sobie dokładniej jak to wszystko będzie przebiegało. Algorytm najpierw renderuje sześć map, które są ścianami sześcianu otaczającymi nasz punkt. Następnie z tych sześciu map są tworzone tekstury, które nakładane są na kulę w specjalny sposób. Zostaną one zmapowane tak, żeby wszystkie sześć zmieściło się na raz na kuli i żeby żadna na inną nie zachodziła. Kolejną rzeczą będzie takie ich nałożenie, aby jedna z nich została umieszczona dokładnie tak, żeby była na wprost naszej kamery, co oczywiście w konsekwencji pociągnie odpowiednie ułożenie pozostałych. Dzieje się tak dlatego, że mapa środowiska zależy od widoku (punktu patrzenia). I tak wyrenderowana nasza kula może posłużyć do stworzenia naszej mapy środowiska. Każda z sześciu map jest w tym wszystkim jakąś częścią naszej kuli. Warto jeszcze zauważyć, że im bardziej szczegółowa będzie kula tym wierniej mapa zostanie odwzorowana - będzie większa możliwość odpowiedniego dopasowania współrzędnych mapowania. W praktyce jednak siatka kuli nie musi być jakaś super doskonała aby skonstruować użyteczną mapę środowiska. Aby wyrenderować naszą kulę musimy przede wszystkim tak ją zorientować, aby poszczególne płaszczyzny ją tworzące odpowiadały nieznormalizowanym wektorom odbić. Każdą taką płaszczyznę należy potraktować w sensie koordynat 2D (**u**, **v**), gdzie **u** i **v** leżą w zakresie od -1 do 1. Dla przykładu rozważmy współrzędne (0.3,-0.4) na płaszczyźnie Y=-1 (przy założeniu, że widoczna powierzchnia kuli jest zorientowana wzdłuż dodatniej osi Z), co będzie oznaczać, że płaszczyzna jest skierowana prosto do obserwatora. Ta płaszczyzna jest kojarzona z wektorem refleksu o współrzędnych (0.3,-1.0,-0.4). Wektor ten należy znormalizować i za jego pomocą obliczyć współrzędne mapowania. Potraktujmy teraz parę **s** i **t** jako pozycje wierzchołków w przestrzeni 2D w przedziałach (-1, 1). Aby poprawnie zmapować teksturę będziemy oczywiście musieli zgodnie ze wzorami podzielić te wartości przez 2 i dodać do tego 1/2. Wtedy współrzędne **s** i **t** będą mieścić się w standardowym przedziale (0, 1). Tych przeliczonych współrzędnych należy użyć jako współrzędnych tekstury dla pozycji wierzchołków w 2D. Następnie należy ustawić ortogonalną macierz projekcji (która nie zmienia wymiarów), określić odpowiednią rozdzielczość naszej mapy (na przykład 128 x 128) oraz ustawić właściwą teksturę dla odpowiedniej płaszczyzny. W ostatnim kroku tworzymy naszą mapę sferyczną używając algorytmu mapowania z sześcianu na płaską teksturę renderujemy naszą mapę przypisując współrzędne tekstury tak jak to opisaliśmy.
W specjalny sposób muszą być obsługiwane tylne powierzchnie naszej kuli. Tekstura nakładana na tylną stronę kuli jak się później okaże będzie pokrywać ją w największej części i możemy sobie wyobrazić, że będzie to coś w formie pierścienia (przy widoku kuli od przodu). W mapowaniu sferycznym centrum tylnej ściany naszej kuli staje się szczególnie ważne przy okrągłej krawędzi naszej mapy sferycznej (bo ta jest de facto jak wiemy w kształcie koła). Najprostszą metodą renderingu tylnej ściany jest podzielenie jej na cztery mniejsze. Drugą sprawą, na którą trzeba zwrócić uwagę to fakt, że jeśli użyjemy pojedynczego wielokąta jako tylną powierzchnię to na obrazie tekstury nasza krawędź nie będzie idealną sferą. Dlatego też dobrze jest aby tylna płaszczyzna kuli była podzielona na drobniejsze elementy na krawędziach koła, co spowoduje, że cała okrągła mapa sferyczna zostanie wyrenderowana poprawnie. Ostatnim krokiem w tej metodzie będzie oczywiście przekopiowanie otrzymanych pikseli na jakąś teksturę i zapisanie jej w pliku.
Przydatną rzeczą podczas obliczania współrzędnych mapowania sferycznego, może się okazać umiejętność odwracania współrzędnych - czyli obliczenia w jakim punkcie na kuli znajdzie się dany piksel mapy środowiska. Jak wiemy tekstura jest płaska i posiada dwa wymiary jeśli popatrzeć na nią z punktu widzenia współrzędnych mapowania - **s** i **t** (albo **u** i **v**). Jak mieliście okazję się przekonać, tekstura zawierająca mapę środowiska zawiera koło, w którym zawartość tej tekstury jest umieszczona; kwadrat tekstury nie jest cały wypełniony. Załóżmy sobie teraz taką rzecz - niech koło, które jest umieszczone na tej teksturze będzie kołem jednostkowym (o promieniu jeden). Każdej współrzędnej **s** i **t** na mapie możemy przypisać miejsce na kuli **P**, z której taka mapa będzie tworzona. Poszczególne współrzędne z 2D mapy będą się przekładać na współrzędne 3D w sposób następujący:

$$P_x = s$$

$$P_y = t$$

$$P_z = \sqrt{1.0 - (P_x)^2 - (P_y)^2}$$

I to w zasadzie byłoby na tyle, co powinniście wiedzieć o środowiskowym mapowaniu sferycznym. Wzory, które podałem na obliczanie współrzędnych na mapie świata oraz sposób ich uzyskania posłużą nam w przykładach do napisania shadera, który będzie umożliwiał uzyskanie efektów refleksu za pomocą mapowania sferycznego. Wprawdzie można zarówno w Direct3D jak i OpenGL uzyskać mapowanie sferyczne w prostszy sposób, ustawiając pewne parametry renderowania, ale my zrobimy sobie to na shaderze - będzie to miało dwie zalety. Pierwsza to oczywiście możliwość ingerencji w sposób przeprowadzenia mapowania (możemy dodać coś od siebie) no a drugi to taki, że będziemy mieli okazję wypróbować shader w ogniu walki i przekonać się czy naprawdę to rozumiemy. A na dziś to byłoby tyle i obiecuję szybką kontynuację. Do zobaczenia!